Integrating Cadence SMV in the Verification of UML Software*

S. Van Langenhove and A. Hoogewijs

Department of Pure Mathematics and Computer Algebra
Ghent University, Belgium
{Sara.VanLangenhove, Albert.Hoogewijs}@UGent.be

The Unified Modeling Language (UML) is a graphical language for modeling a multitude of software-intensive systems. This discussion focuses on the use of Cadence SMV[1] in the design of software systems in UML. We discern two distinct phases in software factoring. In the *design phase* we unveil system properties and inconsistencies by analyzing requirements, and decomposing the system in several communicating components, using standard (and well known to programmers) object oriented reasoning. Inconsistencies and design errors are subsequently removed *before* entering our next phase, the *code generation*. Most contemporary programmers and engineers do not discern such a design phase. We are convinced, as are many researchers, that it is tedious practice to recover from faulty reasoning *during* or *after* the coding phase. Moreover, removing such design errors requires a lot of money and time spent on coding and testing of software. This situation is particularly unacceptable for security, safety or commercially critical systems. Alas, modeling with UML neither guarantees a faultless design nor has the formal backbone required to do so. Therefore, it is important to conceive and to subsequently use a UML based verification method to identify and remove errors, inconsistencies, and misconceptions during the design phase.

A translation template in the Cadence SMV modeling tool for UML state charts is available. This template allows us to convert arbitrary state charts to the modal logic of SMV, covering all behavioral model elements as defined in the UML specification[2], in particular, hierarchy, sequentialism, parallelism, non-determinism, priorities, and run-to-completion step semantics. The SMV model checker is subsequently used to verify the behavioral design. We are also able to verify requirements between communicating objects of different classes using this template. Communicating objects have their behavior specified in separate state charts. The communication is based on signal and call event exchange, also known as asynchronous and synchronous communication respectively, and their queueing. Additionally, the first step towards the protocol conformance between behavioral state charts and protocol state charts has easily been realized.

The template is built using some of the strengths of the Cadence SMV language. Unfortunately, due to some technical limitations and drawbacks of SMV, queuing of events is less trivial as it may seem. The limitations also force us to transform the original state charts into semantic equivalents. More specifically, insertion of additional states and transitions is necessary to cope with these limitations. As a consequence, a number of secondary problems show up. We are forced to perform transformations, using temporal logic, on the system requirements, as specified by the designer. To be able to verify the protocol conformance, protocol state charts have to be transformed too, according to a limited number of rules.

Using the strengths of the Cadence SMV language and after solving the bottlenecks, the template makes way for automatic translation of UML state charts. With such an automatic translation available, UML not only becomes a standard for systems development, but also portal to formal verification. Hence UML is tuned into a more powerful and interesting tool, particularly for engineers and programmers.

*Funded by Ghent University (BOF/GOA project B/03667/01 IV1) and the Prof. Dr. Wuytack Fund

[1] Cadence SMV was developed by K. McMillan and is available at `http://www-cad.eecs.berkeley.edu/~kenmcmil/smv`

[2] Available at `www.omg.org`